LEVEL II ①

AD A101649

NPS55-81-005

NAVAL POSTGRADUATE SCHOOL

Monterey, California

DTIC
ELECTE
JUL 2 1 1981
S D
A

THE NEW NAVAL
POSTGRADUATE SCHOOL
RANDOM NUMBER PACKAGE
LLRANDOMII

by

Peter A. W. Lewis

Luis Uribe

February 1981

Prepared for;

Naval Postgraduate School
Monterey, Ca. 93940

DTIC FILE COPY

81 7 21 057

NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

Rear Admiral J.J. Ekelund
Superintendent

David A. Schrady
Acting Provost

This work was supported in part by the Office of Naval Research under Grant NR-42-284 and in part by the NPS Foundation.

Prepared by:

PETER A.W. LEWIS, Professor

L. Uribe by P.A.W.L.
LUIS URIBE
Department of Operations Research

Reviewed by:

KNEALE T. MARSHALL, CHAIRMAN
Department of Operations Research

Released by:

WILLIAM M. TOLLES
Dean of Research

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: 14 NPS55-81-005

2. GOVT ACCESSION NO.: AD-A101649

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle): 6 The New Naval Postgraduate School Random Number Package-LLRANDOMII.

5. TYPE OF REPORT & PERIOD COVERED: 9 Technical rept.,

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s): 10 Peter A.W. Lewis, Luis Uribe

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Postgraduate School, Monterey, CA 93940

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS: Naval Postgraduate School, Monterey, CA 93940

12. REPORT DATE: 11 February 1981

13. NUMBER OF PAGES: 12 15

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)

15. SECURITY CLASS. (of this report): Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

LLRANDOMII, Psuedo random numbers, order statistics

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes the usage of LLRANDOMII, an IBM 370 set of Assembly Language programs which generate arrays of integer and real (floating point) pseudo random numbers, as well as arrays of Normal, Exponential, Gamma, Cauchy, Poisson and Geometric random deviates. In addition the package will generate order statistics associated with these deviates. A shuffled version of the two basic random number generators in the package can be used for greater statistical reliability. Timings for these very fast subroutines are given, as well as timings for comparable IMSL Chapter G. subroutines.

DD FORM 1473 1 JAN 73 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

251450

TABLE OF CONTENTS

1.0 INTRODUCTION

This Users Guide explains how to use the new version of the Naval Postgraduate School Random Number Generator Package [1], called LLRANDOMII. It is an outgrowth of the old LLRANDOM package [1] which has been the subject of several modifications and enhancements during the period 1978 to 1981, and which was based on the work of Lewis, Goodman & Miller [2].

This document does not go into the details of how the different methods operate internally, but concentrates on helping the reader to be able to use the package. The details are discussed in a separate report [3]; other references are given there. However, the philosophy behind the package is to provide arrays of uniform and non-uniform random numbers which have:

(i) known and documented statistical properties [6]:

(ii) are efficiently computed; and

(iii) are reliably computed.

With point (iii) in mind, nothing in LLRANDOM was changed unless it significantly enhanced the capabilities of the package.

This LLRANDOMII package is mostly written in IBM/360 assembly language but uses two subroutines written in standard IBM FORTRAN. Since IBM/360 Assembly language is upward compatible, the program will run on IBM/370 machines and on the IBM 3033. Testing was done on an IBM/360/65 and on an IBM 3033. All the programs are independent modules which follow all IBM standard linkage conventions and therefore can be called from all IBM high level languages which use the standard linkage conventions (FORTRAN, PL/I, etc.).

The entire package is single precision oriented, so all the parameters are either full word integers or short precision floating point. Beside integer and floating point pseudo-random deviates the package can generate Normal, Exponential, Gamma, Cauchy, Poisson, and Geometric random deviates, or the order statistics associated with these random deviates.

2.0 DIFFERENCES FROM THE PREVIOUS PACKAGE

The new version of LLRANDOM (LLRANDOMII) is not compatible with the old version. Therefore users who may want to convert to the new package should change their subroutine name and argument list since both have changed from LLRANDOM to LLRANDOMII (subroutine names in LLRANDOMII are all different from corresponding subroutine names in LLRANDOM).

The new package has a new parameter (except on integer uniform generators), the fifth one on the argument list, which must be included. This parameter must be either integer 0 to indicate that the array of random numbers is not to be sorted or integer 1 to request that the array of random numbers be sorted in ascending order. Table 1 contains all the subroutine names for all generators available with the argument list included.

The 4th argument in all generators indicates the multiplier to be used in the congruential generator. The table of multipliers currently has two choices (1 = 16807, 2 = 397204094). There are two versions of each generator, the second providing deviates based on a self-shuffled version of the basic random number generator [3].

Using the long multiplier results in "better" random numbers at the expense of an increase in execution time (up to twice as much in an IBM/360. No difference in time found in the IBM 3033. The shuffling feature also improves the quality of the "uniform" random numbers generated, and hopefully improves the quality of the resulting non-uniform numbers generated when that is the case. Shuffling adds only a very small amount of time when used and allows the generation of sequences of uniforms of very long cycle. Given the availability of the long multiplier and the shuffling feature it may have been wise to drop the short multiplier. However this multiplier, originally proposed by Lewis, Goodman & Miller [2], is widely used and is only marginally defective. Thus it can be used to rerun old simulations or to obtain in this package two different pseudo-random number streams.

The new package does not require the initial call to the OVFLOW subroutine to establish the interrupt environment. This is handled internally now by the LLRANDOMII programs, and represents a significant improvement.

New generators have been added for Geometric and Poisson deviates, enhancing the Normal, Gamma, Exponential and Cauchy capabilities of LLRANDOM and its extensions [5]. A new faster algorithm is used to generate Gamma deviates when $\alpha > 1$.

TABLE 1
SUBROUTINE NAMES IN LLRANDOMII PACKAGE

DISTRIBUTION	NOT SHUFFLED	SHUFFLED
UNIFORM (INTEGER)	LINT (IX,A,N,MUL)	SLINT(IX,A,N,MUL)
UNIFORM (REAL)	LRND (IX,A,N,MUL,ISORT)	SRND (IX,A,N,MUL,ISORT)
NORMAL ($\mu=0;\sigma=1$)	LNORM(IX,A,N,MUL,ISORT)	SNOR (IX,A,N,MUL,ISORT)
EXPONENTIAL ($\mu=1$)	LEXPN(IX,A,N,MUL,ISORT)	SEXPN(IX,A,N,MUL,ISORT)
CAUCHY (STANDARD)	LCCHY(IX,A,N,MUL,ISORT)	SCCHY(IX,A,N,MUL,ISORT)
GAMMA ($\mu=1;\alpha>0$)	LGAMA(IX,A,N,MUL,ISORT,ALFA)	SGAMA(IX,A,N,MUL,ISORT,ALFA)
POISSON ($\lambda>0$)	LPOIS(IX,A,N,MUL,ISORT,AL)	SPOIS(IX,A,N,MUL,ISORT,AL)
GEOMETRIC ($0<P<1$)	LGEOM(IX,A,N,MUL,ISORT,P)	SGEOM(IX,A,N,MUL,ISORT,P)

DEFINITIONS:

- IX = Seed. Full word integer.
- N = Number of deviates to generate. Full word integer.
- A = Array of size $\geq$ N to store deviates. Full word integer in the case of Uniform integers. Short precision floating (Real *4) otherwise.
- AL = Parameter LAMBDA of Poisson generators. Short precision floating point.
- P = Parameter P of Geometric generators. Short precision floating point.
- ALFA = Shape parameter ALFA of Gamma generators. Short precision floating point.
- ISORT = 0 for no sort. 1 for sorted deviates. Full word integer.
- MUL = Multiplier to be used. At this time multipliers M1 = 16807 and M2 = 397204094 are implemented (MUL = 1 gives M1; MUL = 2 gives M2).

3.0 RELATIONSHIP TO IMSL (EDITION 8), CHAPTER G

The routines in the International Mathematics and Statistics Libraries (IMSL) Edition 8 (designed by P.A.W. Lewis & J. Gentle) implement substantially the same algorithms as are implemented in LLRANDOMII. However the implementation in the IMSL Library is meant first of all to be portable across the computing systems which IMSL supports, and secondly is oriented toward generation of random numbers one at a time. Thus for efficiency the LLRANDOMII package should be used.

Timings for the LLRANDOMII routines are given in Table 2 below for the IBM 3033. Some timings are given for IMSL subroutines. In all cases timings per deviate are based on generation of arrays of 10,000 deviates. Timings for LLRANDOM were given in [1]. LLRANDOMII on an IBM 3033 is roughly eight times as fast as LLRANDOMII on the IBM 360/67.

TABLE 2

TIMING COMPARISONS AND SUBROUTINE EQUIVALENCES

VARIATE	LRANDOMII IBM 3033			IMSL IBM 3033		
INTEGER UNIFORM	LINT		(2)	GGUD		(16)
REAL UNIFORM	LRND		(3)	GGUBS		(9)
NORMAL	LNORM		(7)	GGNML		(44)
EXPONENTIAL	LEXPN		(7)	GGEXN		(23)
GAMMA (CHI-SQUARE)	LGAMA	$\alpha = .8$	(54)	GGAMR	$\alpha = .8$	(91)
		$\alpha = 2.5$	(18)		$\alpha = 2.5$	(105)
GEOMETRIC	LGEOM	$p = .8$	(5)	GGEOT	$p = .8$	(21)
		$p = .98$	(16)		$p = .98$	(21)
POISSON	LPOIS	$p = .8$	(5)	GGPON	$p = .8$	(29)
		$p = 20.$	(7)		$p = 20.$	(323)
CAUCHY	LCCHY		(7)	GGCAY		(34)

- Time in microseconds (10^{-6} secs) per deviate based on generation of arrays of 10,000 deviates at a time is given in parenthesis. The Assembly Language LLRANDOMII routines are usually 3 to eight times as fast as the FORTRAN IMSL subroutines.
- For times to generate deviates one at a time add approximately 7 microseconds to the values for LRANDOMII subroutines, 12 to 20 microseconds to IMSL, subroutines.
- Use of the shuffled generators adds about 5% to the times for LRANDOMII subroutines.

4.0 USING THE PACKAGE

For most of the uses initialization of the "seed" IX is the only thing required in addition to the call to the appropriate subroutine. There are some cases though when the user may want to generate several streams of numbers and then he must be aware of some rules to follow if he is concerned about being able to repeat the exact sequence of numbers later on. The "seed" is the starting value for the (integer) uniform random number generator and it must be initialized to any value between 1 and $2^{31}-1$. It is updated automatically on every call to the generators so that next time the subroutine is called the sequence of numbers produced will continue at the point it ended last time. Normally, the user should never change the seed in the course of the program. If the user wants different streams of numbers at different points in his program, then he should use different seed names and initialize them with different values for the different calling points.

In the new version of LLRANDOM all the generators are independent of each other since each generator is now an independent module with its own separate storage. However, using the same generator at different points in the same program will still cause some interaction in the case of "shuffled" generators. This is because only one copy of the generator is loaded by the link-editor and the shuffling table therefore becomes the same for the different calling points. Then, in this case, if repeatability is important the user must make sure that the "seeds" and the number of deviates requested at every calling point remain the same. Section 4.1 shows an example of a case where the sequences do not match in different runs of the program due to a "small" modification made to it. As indicated before this happens only when using shuffled generators.

4.1 EXAMPLES

a) To generate 10,000 geometric deviates with parameter p=0.80, using the short multiplier, no shuffling, no sorting. The Geometric generator produces deviates X according to the following form of the Geometric distribution:

$$f(k)=P\{X=k\}=p^k(1-p), \qquad k=0,1,\ldots,$$

which has moments

$$E(X)=p/(1-p), \qquad var(x)=p/(1-p)^2.$$

The FORTRAN program may look like this

```
REAL *4 A(10000), P
Integer IX,N
IX=15987
N=10000
P=.80
:
Call LGEOM (IX,A,N,1,0,P)
:
End.
```

Thus there is no sorting and the Multiplier 1 is used.

b) Generate two batches of ten samples each of Poisson deviates. Batch No. 1 with samples of size N1=5000 each and Batch No. 2 with samples of size N2=3500 numbers. Both batches are from the Poisson distribution with $\lambda = 3.7$, using multiplier 2, shuffled and no sorting.

The Poisson distribution has the form

$$f(k)=P\{x=k\}=\lambda^k e^{-\lambda}/(k!)$$

with moments

$$E(x)=\lambda, \qquad var(x)=\lambda.$$

The FORTRAN program may be set up like this:

```
         Real *4 A1(5000),A2(4000), AL
         Integer IX1, IX2, N1, N2
         IX1=14872
         IX2=932789
         N1=5000
         N2=3500
         AL=3.7
         .
         .
         .
         Do 100 ISAMPL=1,10
         .
         .
         .
c    Calling sequence for samples of Batch No. 1
          CALL SPOIS (IX1,A1,N1,2,0,AL)
         .
         .
         .
c    Calling sequence for samples of Batch No. 2
          CALL SPOIS (IX2,A2,N2,2,0,AL)
         .
         .
         .
100      CONTINUE
         .
         .
         .
         END
```

This example shows one case where the same shuffled generator is used in different points in the same program, and interaction between the two calls is created. If the same sequences of numbers have to be repeated in a future run of the program, the user must make sure that IX1,IX2, N1 and N2 remain the same. For example, if the program is run again with N2=4000 the shuffling table (which is shared by the two calls) will be updated more often now in the second call causing the numbers from the first call to be different.

As indicated before this happens only when the same subroutine name is used in both calls and they are of the shuffled type.

As far as the no-shuffle type generators being able to regenerate the same sequences of numbers, all that is required is to use the same value for the "seed" in all different runs of the program. Different names for the seed variable are still necessary to avoid interaction between the different calls, in the case of multiple calls in the same user program.

4.2 NOTE FOR PL/1 USERS

When using the LLRANDOMII package from PL/1 programs the appropriate subroutines being used must be declared as external entries with data types Binary Fixed (31) and Binary Float (21) for the parameters to be passed that correspond, respectively, to INTEGER and REAL *4 in Fortran. For example, if the LGAMA generator is used in a PL/1 program, the following declaration must be included in the program:

```
DCL   LGAMA EXTERNAL ENTRY (BIN FIXED(31), (*) BIN FLOAT(21),
      BIN FIXED(31), BIN FIXED(31), BIN FIXED(31), BIN FLOAT(21))
                                           OPTIONS(ASSEMBLER INTER);
```

5.0 AVAILABILITY

The LLRANDOMII package is distributed in 9 track 1600BPI magnetic tape containing two partitioned data sets. The first contains the source IBM Assembly and Fortran code and the second contains object code that can be used to construct a TXTLIB under VM/CMS or as a library in the link-edit step when used in batch mode.

To obtain a copy send a scratch tape to

Naval Postgraduate School
Monterey, CA 93940
Operations Research Dept.
Attn: Professor P.A.W. Lewis

6.0 REFERENCES

[1] Learmonth, G.P. and P.A.W. Lewis, "Random Number Generator Package", Naval Postgraduate School Report NPS55LW73111A, Naval Postgraduate School, November 1973.

[2] Lewis, P.A.W., A.S. Goodman and J.M. Miller, "Pseudo-Random Number Generator for the System/360," IBM Systems Journal, No. 2, 1969.

[3] Lewis, P.A.W. & Uribe, L., "The New LLRANDOMII Random Number Package" To appear as NPS Technical Report.

[4] Learmonth, G.P., "Empirical Tests of Multiplier for the Prime Modulus Random Number Generator $X_{i+1} \equiv AX_i \pmod{2^{31}-1}$", Naval Post Graduate School Report NPS55-77-30, Naval Postgraduate School, June 1977.

[5] Robinson, D.W. & Lewis, P.A.W., "Generating Gamma & Cauchy Random Variables: An extension to the Naval Postgraduate School Random Number Generator Package," 1974.

[6] Learmonth, G.P. & Lewis, P.A.W. "Statistical Tests of Some Widely Used and Recently Proposed Uniform Random Number Generators," Naval Postgraduate School Report NPS55LW73111A.

INITIAL DISTRIBUTION LIST

	NO. OF COPIES
Defense Techincal Information Center Cameron Station Alexandria, VA 22314	2
Library, Code 0142 Naval Postgraduate School Monterey, CA 93940	2
Library, Code 55 Naval Postgraduate School Monterey, CA 93940	1
Dean Research Naval Postgraduate School Code 012A Monterey, CA 93940	1
Statistics and Probability Program Code 436, Attn: E.J. Wegman Office Of Naval Research Arlington, VA 22217	3
Office of Naval Research Probability and Statistics Lists; Basic	37
Office Of Naval Reserach Probability and Statistics List; Mathematical and Computational	44
Office of Naval Research Simulation List	67
Attn: P.A.W. Lewis, Code 55Lw	1000

DISTRIBUTION LIST

	No. of Copies
Naval Postgraduate School Monterey, CA 93940	
Attn: Code 55Mt	1
Code 55As	1
Code 55Bn	1
Code 55Bw	1
Code 55Cu	1
Code 55Ei	1
Code 55Ey	1
Code 55Fo	1
Code 55Gv	1
Code 55Hh	1
Code 55Hk	1
Code 55Hl	1
Code 55Jc	1
Code 55La	1
Code 55Lw	1
Code 55Ls	1
Code 55Mg	1
Code 55Mh	1
Code 55Mu	1
Code 55Mp	1
Code 55Ni	1
Code 55Py	1
Code 55Pk	1
Code 55Re	1
Code 55Rh	1
Code 55Ro	1
Code 55Sy	1
Code 55Su	1
Code 55Ta	1
Code 55Tw	1
Code 55Ty	1
Code 55Ws	1
Code 55Ze	1
L. Ishii	1
Code 52Bz	1